# The Warplan: A Method Independent Plan Schema

Adam Pease
Teknowledge
1810 Embarcadero Rd
Palo Alto, CA, 94303
apease@teknowledge.com

**Abstract**

The Core Plan Representation (Pease 1997) is a schema for the representation of plan information that is independent of any particular method of plan construction. It has been developed as an ontology and implemented as an object oriented design. Now in its second phase, the generic plan design has been specialized for several areas of military planning – the Warplan. This specialization and implementation is now undergoing examination as a possible foundation for several new DARPA projects.

## Background

The Core Plan Representation (CPR) (Pease & Carrico 1997) is an effort to develop a plan ontology that supports the representation needs of many different planning systems. It has been developed in part for the Joint Task Force Advanced Technology Demonstration (JTF-ATD) (Hayes-Roth, 1995). The goal of this effort is to leverage common functionality and facilitate the reuse and sharing of information between a variety of planning and control systems. The CPR is a standard that is general enough to cover a spectrum of domains from planning and process management to workflow and activity models. In addition, the proposed representation will be powerful enough to support complex, hierarchical plan structures.

The design of the CPR is an attempt to unify the major concepts and advancements in plan and process representation into one comprehensive model. The problem of planning, and plan representation has been an ongoing area of active research. This paper presents the current state of the Core Plan Representation and shares some of the experiences of trying to draw many successful domain efforts into a single, unified, method independent design.

There are two significant payoffs to the CPR effort. The first is that creation of a base plan representation will facilitate information interchange among different planning systems. Imagine a typical military planning situation. A crisis develops and a joint task force is formed. The leadership and staff use a planning application to develop guidance for their subordinate commands. This guidance includes background on the situation, objectives that must be met to contain the crisis, constraints on the actions of the task force and high level specification of the schedule of operation. This information is passed to individual commands that have specific requirements and methods of planning. A standard plan framework enables improved information transfer to these specialized planning applications. Continuing to follow this generic and hypothetical example, the commander of the air component of the task force and his staff will use their superiors' objectives to develop more detailed objectives, lists of targets which support those objectives, and then repeatedly create a schedule of aircraft sorties to destroy those targets. Pilots flying those sorties could benefit from performing simulated runs. A core plan representation enables information from the plan to be transferred to simulation entities possibly allowing a single pilot to fly along side computer generated forces simulating the other pilots in his flight. While information transfers of this sort will rarely be complete, and will often require further augmentation and elaboration for the new application, the CPR will reduce the amount of manual rekeying and reformulation of existing data. The Warplan is an effort to specialize the CPR and make this hypothetical example a reality

The second payoff is in the creation of common services based on the CPR. There are two broad areas of services with immediate utility. The first area is visualization. Manufacturing, business planning and construction management all share several basic forms of visualizing plan information. The CPR enables creation of these common views that will dramatically reduce implementation time for specific systems. Well-designed common viewers can then be specialized for particular planning applications instead of written from scratch. The second area is scheduling. Many important advances have occurred in the operations research and artificial intelligence communities that allow software systems to provide significant aid in complex scheduling problems. A complete scheduling system can rarely be build on generic techniques alone, but the CPR enables the creation of generic scheduling tools which eliminate the need to build these tools from scratch each time a new scheduling domain is targeted. An implementation of the CPR is now underway. It incorporates a simple constraint resolver, views of the tree structure of a composite plan and a Gantt chart for viewing of temporal and precedence relationships between actions.

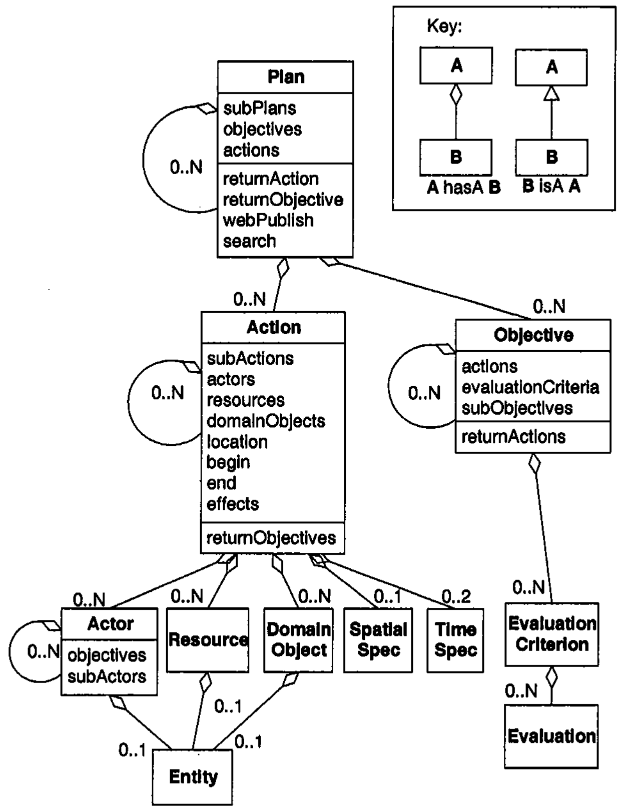

**Figure 1 - CPR Partonomy**

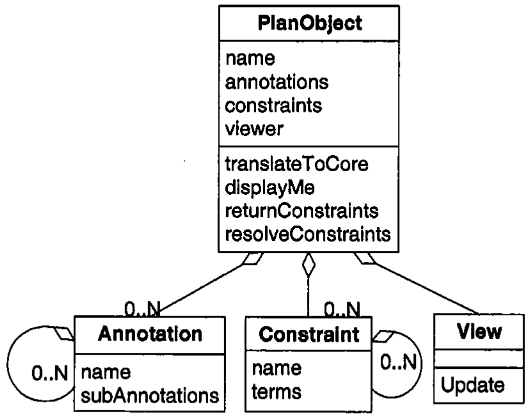

**Figure 2 - CPR Common Superclass**

## CPR Concepts

A *Plan* consists of one or more *Actions* performed in pursuit of some *Objective*. *Actors* may use *Resources* in performing an *Action*. *Actor* and *Resource* are not directly elements of the *Plan*, but rather of the *Action* itself.

*TimeSpec* represents some association of the *Action* with time. Though a number of *TimeSpecs* may provide information about an *Action*, every *Action* has at least a beginning and end, though either may be infinite or periodic. In considering the temporal reference of the *Action*, it is only natural to then consider the relevance of the spatial references. A *SpatialSpec* is added meet this need. As with *TimeSpec*, *SpatialSpec* may represent an exact location, or a vague area. It may ground the objects both in absolute and relative terms. *SpatialSpecs* are also associated with *Actions*. Every *Action* takes place in some space whether the space is real or virtual.

*Constraints* need to be added also. *Constraints* are restrictions on other elements of the plan. For example, we might need to specify that one *Action* must take place in a certain proximity to another *Action*, or that the end *TimeSpec* of one *Action* occurs before the begin *TimeSpec* of another. *Constraints* have no single place in the CPR where they would be most appropriate. As such, it is left as an object that can be contained in any object of the plan.

Due to aggregate nature of *Plans*, it is appropriate to allow a *Plan* to be associated with another *Plan*, where one would be the parent plan and the other a sub-plan. A similar argument may be made for both *Objectives* and *Actions*, representing sub-objectives and sub-actions respectively.

During execution of the *Plan*, *Objectives* are reviewed in order to gauge the effectiveness of the *Plan* and identify when the *Plan* is complete. This review is performed against a set of evaluation criteria relevant to the *Objective*. The entity *EvaluationCriterion* is added to *Objective* meet this need.

## CPR Attributes

The next step is to add attributes to the classes. Attributes that are plural may be understood as containing a list of objects.

First, *Plan* needs to be completed. We add the attributes *subPlans*, *actions*, and *objectives*. *Action* may now be elaborated. We have already identified *subActions*, *actors*, *resources*, and *begin* and *end TimeSpecs*. *DomainObjects* are defined to represent entities referred to in an *Action* which are not the *Actor* or a *Resource*. For example, the recipient of a mail message would be recorded as an instance of *DomainObject*. We also define *effects* to state how an *Action* is expected to change the world

It is clear that most plan objects require a name. In addition, it is valuable to be able to add *Annotations* to any element of the plan. *Constraints* may also apply to any plan object. For this reason, the common superclass of *PlanObject* is created containing the attributes of *annotations*, *constraints*, and *viewer*. This class is defined as the common superclass for all the objects in Figure 1.

*Actors* may have objectives of their own and so a reference to the plan objectives is added and called *objectives*. Since the *Actor* may not be a single person, but represent an aggregate of some kind, the attribute *subActors* was added.

*Objective* contains *subObjectives*, and *evaluationCriteria*. *Objective* also references the list of

*Actions* in the *Plan* which are meant to satisfy it. This information is held in the attribute *actions*.

In order to complete *Constraint*, additional fields are added to hold the *terms* of the constraint expression, and any associated *subConstraints*. These are represented as attributes of type *Term* and *Constraint*.

*Annotations* consist of a *name* and the body of the annotation. In addition, the annotations could be constructed hierarchically to form linked documentation, and thus a list of *subAnnotations* is required.

## CPR Functions

A few generic functions are included in CPR to ensure that specializations can function effectively with tools that only assume the presence of the core. One group of functions are those which simply return a set of objects. `returnAction`, `returnObjective` and `returnConstraints` provide a common way to return objects in the plan without regards to the structure of the objects which may contain them. They also provide for a mechanism by which "virtual" objects may be returned. For example, it is quite common in plans for a series of *Actions* to be executed sequentially. CPR would require *Constraints* between the *TimeSpecs* of each *Action* enforcing sequentiality. However, *Action* could be specialized into a *SequentialAction* which simply knew to generate the implied sequentiality *Constraints* in response to a `returnConstraints` message.

`resolveConstraints` provides an interface to constraint resolution functions. `webPublish` suggests a method which streams the entire plan into an HTML file for viewing. `search` ensures that all specializations provide at least one way of searching for objects in the plan structure. `displayMe` provides a way to activate a window which is a view on the information contained in the object sent this command. `translateToCore` ensures that all specializations provide a mechanism to convert themselves to use only those objects specified in the CPR. For example, specialized *Actions* with implied *Constraints* would have to return a structure that made those *Constraints* explicit.

## Related Research

This work has been influenced by many other efforts, a review of which is outside the scope of this document. A reader who is interested in the broad history of planning and plan representation is referred to (Tate et al 1990), (Allen et al 1990), and (Zweben & Fox 1994).

Several efforts have significantly influenced the CPR. The planning representation of KRSL (Lehrer 1993) bears many similarities to CPR. In fact, some of the same people working on KRSL-Plans have contributed significantly to the CPR design. The presentations of the two efforts are slightly different however. While KRSL-Plans presents an ontology in informal sentences, CPR presents an object oriented software design developed from an ontology. It should be noted that KRSL-Plans, like CPR, is also an ongoing effort, although it began much earlier. KRSL also attempts a more exhaustive cataloguing of domain specific terms than CPR without further definition. <I-N-OVA> (Tate 1996) is another ongoing effort that bears some similarity to CPR due to the significant contribution of Austin Tate to the CPR effort. Another important related effort is the Process Interchange Format (Lee 1996).

Many projects are now converging. The NIST Process Specification Language (Schlenoff, et al 1997) is an effort in plan representation for manufacturing. The CPR has been one of the inputs to this representation. The DARPA Shared Planning and Action Representation (SPAR 1997), (Tate 1998) has been an additional effort by the DARPA research community in this area. CPR is the object oriented representation and reference implementation for SPAR. Under the DARPA High Performance Knowledge Bases project, CPR will be part of the Teknowledge-Cycorp knowledge base derived from Cyc (Lenat 1995).

## Warplan Specialization

The Warplan adds three specializations to this core: Intelligence Planning, Operations and Logistics.

### Operations

This design specializes the CPR for military operations planning. The first step was to harmonize the design with a grammar for Air Campaign Planning Objectives (Valente 1996). Sixteen objects specific to ACP were added to CPR. This harmonization was quite interesting for several reasons. First, the ACP grammar was written in LOOM (MacGregor, 1994) and the CPR uses UML (Rational, 1997). Harmonizing such different paradigms of knowledge coding required considerable explanation by each author.

Our process began with familiarizing ourselves with each other's system. This surprisingly proved to be one of the hardest tasks. While each has been clearly and extensively documented, the meaning of concepts and relations proved elusive. While many formalizations are capable of expressing meaning adequately, questions of context and use and the assumptions that underly those questions can overshadow the care taken in description and explanation. For this reason, some researchers have argued that harmonizing ontologies is an excellent way to debug them in that assumptions are often brought out into the open in such a process.

The words used to embody concepts are necessarily ambiguous. The expressiveness of English carries with it a lack of precision. It is seductive to assume that we know what a system designer meant by a concept just because of its name.

Some researchers have suggested that modern knowledge representations are fully capable of capturing the meaning or conceptualization behind a labeled concept.

While we doubt that claim itself, a stronger form of expression is needed. Even if it were possible to encode meaning in a computable format, it is possible to do so only with incredible effort. It is the relation of a concept to many other concepts and the use of the concept in reasoning that gives it its meaning. Humans communicate in part because a vast web of relations are present in which concepts are embedded. Only through creating such a web will shared meaning be possible.

For example, a fundamental concept in the ACP grammar is the concept "objective". Objectives are a principal way in which commanders communicate with subordinates. They specify what the subordinate is to do and what restrictions might be placed on how he or she must carry it out such as timing, location and resources.

The CPR also has a concept "objective". It is defined as "what is to be achieved" and is disjoint from the concept of "action" which specifies "how to achieve it". Actions have restrictions on time, location and resources.

An initial look at the two representations showed them to be fundamentally incompatible. It was only after considerable discussion that it became apparent that if the terms "action" and "objective" were removed and replaced with empty symbols, then the relations surrounding the symbols mapped perfectly. The meaning was contained in the relations. Once the obstacle of English concept names was removed it was clear that the CPR Action was the same as the ACP Objective.

Given that we had now come to terms with the concepts and their names, there remained the task of conforming to the aesthetics of two different formalisms. A few broad generalizations can be made about each that once understood, take care of many apparent differences. Many are also differences of degree rather than kind.

Object designs favor simple taxonomies which are broad and shallow. Even most proponents of multiple inheritance agree that it should be used sparingly. By contrast, ontologies often have very deep taxonomies and multiple inheritance does not face the same caution.

Object design supports turning complex and important relation into first class objects. The major object oriented (OO) methodologies such as UML support early stages of this design process as well by supporting named links in addition to first class objects. There is however a sense that this feature should be used sparingly - that each object should still have state and behavior. An ontology has no such restriction. In fact, many modern KR languages allow for relations to acquire attributes without further modification.

Overall, for efficiency both of execution and of design, OO designers are much more sparing with the creation of objects. Simple related concepts are often bundled together in the same object if possible. Ontology designers attempt to make all useful distinctions in separate concepts.

After adding the ACP grammar, it was necessary to further specialize the design for an air force planning application called JFACC (Logicon, 1997). Thirty-one objects specific to JFACC were added to CPR. Because the harmonization with ACP had already taken place and because the JFACC design was also an object oriented design, the harmonization with JFACC was much easier than with ACP.

### Intelligence Planning

The Intelligence portion of the Warplan is a harmonization of CPR with the work of (Albericci 1997). This work adds thirty-one objects to CPR. Significant areas of addition include subclassing of Action to cover the five phases of intelligence operations: direction, collection, processing, production and dissemination. In addition, significant subclasses were created for resource types and roles. Roles in particular are significant for the intelligence area. CPR follows a design choice of reifying roles when they are present. While it would be possible to make roles attributes of the entities that play a role, creating separate concepts that relate that entity to a role object is very powerful. Methods that operate on roles that are filled by different entity types can be made more general. By localizing information about the role in its object rather than duplicating that information across entities, modularity and reuse are improved. Another option would be to utilize multiple inheritance but that would restrict the implementation language. Since Java is the current choice for implementation, that option was not feasible.

### Logistics

This work adds detail about logical and physical resources to CPR. It is based on the work of (Kumara 1997), (Smith S.1996), and (Smith, D. 1996). The most significant specializations are an elaboration of Resource types. The first tier of specializations divides resources into a number of mathematically or logically disjoint types: ConsumableResource, ReusableResource, Synchonously-ReusableResource, ExactCapacityResource, and NonShareableResource. The next tier of specializations relate some real world entities to their roles. These include TransportationNode, Route, Transport and Driver-Pilot.

## Implementation

The CPR design is now being implemented. There are several goals to the effort. The first is to begin to make available a reusable planning object library to the DARPA community. By providing a set of tools that manipulate the representation, we offer users a greater benefit than simply providing a design on paper. By providing tools developed with an "open systems" philosophy, it is hoped that many groups can eventually contribute to its evolution. This distributed development should greatly enhance the ability of DARPA planning programs to interoperate and create synergies that were not previously possible.

The second goal is to debug the representation. While the design of CPR appears relatively stable after a year and a half of work and several specialization exercises, it is quite possible that further refinements remain. By distributing development to many groups, it is hoped that the refinement will be rapid and productive.

## Conclusions and Future Work

The domain independent and planning method independent Core Plan Representation has proven to be an effective basis for several domain dependent specializations. Future work includes extending the current CPR implementation to cover the design of the Warplan. Additionally, many generic methods need to be added to the implementation. These include generative and case-based planning. It is hoped that the effort to add different planning methods will help to verify that the design is independent of planning method.

### Acknowledgement

This work was performed under contract to the DARPA Information Systems Office. The author would like to thank Todd Carrico for his continued support and participation in this work.

## References

Albericci, D., 1997, C2 Intelligence Schema Interim Report, Teknowledge, Palo Alto, CA, September 30.

Allen, J., Hendler, J., and Tate, A., 1990. Readings in Planning, Morgan Kaufman.

Hayes-Roth, F. 1995. The JTF ATD architecture for agile, mobile, collaborative crisis response: A case study in architecture-driven development. In *Proceedings of the First International Workshop on Software Architecture.* Pittsburgh, PA: Carnegie-Mellon Univ

Kumara, S., 1997, An Approach to Implement ALP Storyboard: Version 1.0, Pennsylvania State University, University Park, PA, 3 June.

Lee, J. (editor) Micheal Grunninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost and other members of the PIF Working Group, 1996. The PIF Process Interchange and Framework Version 1.1, MIT Center for Coordination Science, Working Paper #194.

Lehrer, N. (ed.), 1993. "ARPI KRSL Reference Manual 2.0.2", ISX Corporation. http://isx.com/pub/ARPI/ARPI-pub/krsl/krsl-info.html

Lenat, D., (1995), Cyc: A large-scale investment in Knowledge Infrastructure, C. ACM, 38, 11, November.

Logicon, 1997, JFACC TIED Program (Phase 2) Object Schema - Draft, Logicon, San Pedro, CA, May 9.

MacGregor, R., 1994. A Description Classifier for the Predicate Calculus, in Proceedings of the Twelfth National Conference on Artificial Intelligence, (AAAI 94), pp. 213-220.

Pease, A. 1997. Core Plan Representation web pages. http://www.teknowledge.com/CPR2

Pease, A. & Carrico, T. 1997. The JTF ATD Core Plan Representation: Request for Comment. Armstrong Lab: AL/HR-TP-96-9631.

Pease, A. & Carrico, T. 1997. The JTF ATD Core Plan Representation: A Progress Report, Proceedings of the AAAI Spring Symposium on Ontological Engineering, Stanford, CA, March, (AAAI Press).

Rational, 1997, UML Summary, Rational Software Corporation, http://www.rational.com/uml/summary/

Schlenoff, C. (ed.), Knutilla, A., and Ray, S., 1996. Unified Process Specification Language: Functional Requirements for Modeling Processes", National Institute of Standards and Technology, Gaithersburg, Maryland. http://www.nist.gov/psl/

Smith, D., 1996, Unpublished slide entitled Resource Types, Kestrel Institute, Palo Alto, CA.

Smith, S. & Becker, M., 1996, An Ontology for Constructing Scheduling Systems, Proceedings of the AAAI Symposium on Ontological Engineering, Stanford, CA, March, (AAAI Press).

SPAR (1997), Shared Planning and Action Representation, http://www.aiai.ed.ac.uk/~arpi/APR/

Tate, A., Hendler, J., & Drummond, M., 1990. A Review of AI Planning Techniques. In *Readings in Planning*, pp. 26-49, (Allen, Tate and Hendler, eds), Morgan Kaufmann.

Tate, A. 1996. Representing Plans as a Set of Constraints - The <I-N-OVA> Model. In *Proceedings of the Third International Conference on Planning Systems* ed. B. Drabble, AAAI Press.

Tate, A. 1998, Roots of SPAR - Shared Planning and Activity Representation, to appear in Knowledge Engineering Review, Special Issue on Ontologies, Cambridge University Press.

Valente, A., 1996, A Representation and Library for Objectives in Air Campaign Plans, USC Information Sciences Institute, Marina del Rey, CA, July 2.

Zweben, M., and Fox, M., 1994. Intelligent Scheduling, Morgan Kaufman.